

Effective on 12/08/2004.
Fees pursuant to the Consolidated Appropriations Act, 2005 (H.R. 4818).

FEE TRANSMITTAL

For FY 2005

☐ Applicant claims small entity status. See 37 CFR 1.27

TOTAL AMOUNT OF PAYMENT (\$) 130

Complete if Known

Application Number	10/801,898
Filing Date	March 15, 2004
First Named Inventor	Yagawa, Yuichi
Examiner Name	Mano Padmanabhan
Art Unit	2188
Attorney Docket No.	16869B-098200US

METHOD OF PAYMENT (check all that apply)

☐ Check ☐ Credit Card ☐ Money Order ☐ None ☐ Other (please identify): _____
☒ Deposit Account Deposit Account Number: 20-1430 Deposit Account Name: Townsend and Townsend and Crew LLP

For the above-identified deposit account, the Director is hereby authorized to: (check all that apply)

☒ Charge fee(s) indicated below ☐ Charge fee(s) indicated below, except for the filing fee
☒ Charge any additional fee(s) or underpayments of fee(s) under 37 CFR 1.16 and 1.17 ☒ Credit any overpayments

WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038

FEE CALCULATION

1. BASIC FILING, SEARCH, AND EXAMINATION FEES

Application Type	FILING FEES Small Entity		SEARCH FEES Small Entity		EXAMINATION FEES Small Entity		Fees Paid (\$)
	Fee (\$)	Fee (\$)	Fee (\$)	Fee (\$)	Fee (\$)	Fee (\$)	
Utility	300	150	500	250	200	100	
Design	200	100	100	50	130	65	
Plant	200	100	300	150	160	80	
Reissue	300	150	500	250	600	300	
Provisional	200	100	0	0	0	0	

2. EXCESS CLAIM FEES

Fee Description	Small Entity	
	Fee (\$)	Fee (\$)
Each claim over 20 or, for Reissues, each claim over 20 and more than in the original patent	50	25
Each independent claim over 3 or, for Reissues, each independent claim more than in the original patent	200	100
Multiple dependent claims	360	180

Total Claims	Extra Claims	Fee (\$)	Fee Paid (\$)	Multiple Dependent Claims	Fee (\$)	Fee Paid (\$)
_____ -20 or HP = _____	x _____	= _____				
HP = highest number of total claims paid for, if greater than 20						
Indep. Claims	Extra Claims	Fee (\$)	Fee Paid (\$)			
_____ -3 or HP = _____	x _____	= _____				
HP = highest number of independent claims paid for, if greater than 3						

3. APPLICATION SIZE FEE

If the specification and drawings exceed 100 sheets of paper, the application size fee due is \$250 (\$125 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).

Total Sheets	Extra Sheets	Number of each additional 50 or fraction thereof	Fee (\$)	Fee Paid (\$)
_____ - 100 = _____	/ 50 = _____	(round up to a whole number) x _____	= _____	

4. OTHER FEE(S)

Non-English Specification, \$130 fee (no small entity discount)

Other: Petition Fee

Fees Paid (\$)

130

SUBMITTED BY

Signature		Registration No. (Attorney/Agent) 37,478	Telephone 650-326-2400
Name (Print/Type)	George B. F. Yee		Date August 3, 2005

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to:

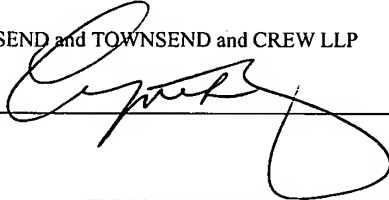
PATENT
Attorney Docket No.: 16869B-098200US
Client Ref. No.: HAL-ID 288

On

8/3/05

TOWNSEND and TOWNSEND and CREW LLP

By:



IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:

Yuichi Yagawa

Application No.: 10/801,898

Filed: March 15, 2004

For: Long Term Data Protection System
and Method

Customer No.: 20350

Confirmation No. 3352

Examiner: Mano Padmanabhan

Technology Center/Art Unit: 2188

PETITION TO MAKE SPECIAL FOR
NEW APPLICATION PURSUANT TO
37 C.F.R. § 1.102(d) &
M.P.E.P. § 708.02, Item VIII,
ACCELERATED EXAMINATION

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Submitted herewith is a petition to make special the above-identified application in accordance with MPEP § 708.02, Item VIII, accelerated examination. The application has not received any examination by the Examiner.

(A) The Commissioner is authorized to charge the petition fee of \$130 under 37 C.F.R. § 1.17(h), and any additional fees that may be associated with this petition may be charged to Deposit Account No. 20-1430.

(B) All the claims are believed to be directed to a single invention. If the examiner determines that all the claims presented are not obviously directed to a single invention, then Applicant will make an election without traverse as a prerequisite to the grant of special status where the specific grouping of claims will be determined by the examiner.

08/08/2005 CNGUYEN2 00000060 201430 10801898

01 FC:1464

130.00 DA

(C) A pre-examination search was performed by an independent patent search firm. The pre-examination search includes a classification search, a computer database search, and a keyword search. The classification search covered the following classes and subclasses:

Class	Subclass
707	10, 202, 204, 205
709	215, 221, 223, 238
710	74
711	117, 141, 153, 161, 162, 170, 173
714	6, 8

Additionally, a keyword search was performed on the USPTO full-text database, including published applications. The following references were identified in the search report:

(1) U.S. Patent Nos.:

US 5,440,727	Bhide et al
US 5,778,395	Whiting et al
US 5,815,649	Utter et al
US 6,192,472	Garay et al
US 6,367,029	Mayhead et al
US 6,405,315	Burns et al
US 6,654,771	Parham et al

(2) U.S. Patent Application Publication Nos.:

US 2002/0032691	Rabii et al
US 2002/0194209	Bolosky et al
US 2003/0200207	Dickinson

(D) The above references are enclosed herewith, collectively as Exhibit A.

(E) Set forth below is a detailed discussion of the references, pointing out with particularity how the claimed subject matter recited in the claims, amended according to the preliminary amendment filed herewith, is distinguishable over the references.

Claimed Subject Matter of the Present Invention

There are nine independent claims among the 50 claims that are pending in the instant application.

Independent **claim 1** is directed to a method of accessing a storage system. The method includes accessing a data object where the data object is divisible into one or more partitions. For each partition, if there are no other partitions among other data objects in the storage system that are identical to that partition, then one or more replicas of the input partition are produced.

Independent **claim 8** is directed to a method of accessing a storage system, including receiving data for a first file, where some of the data constitutes a first partition of the first file. A second partition is defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition. If the number of such second partitions is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. If the number of second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. This is repeated for other partitions which constitute the first file.

Independent **claim 18** is directed to a method for accessing a storage system, including storing a file in the storage system and identifying one or more partitions (referred to as input partitions) of the file. For each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system. An identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

Independent **claim 28** is directed to a data storage system comprising a storage component and a data processing component. The data processing component is configured to access a first partition of a file. If the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

Independent **claim 33** is directed to a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

Independent **claim 35** is directed to a method for accessing a storage system. A first read-out partition of a data object is accessed. If content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. Content constituting the first read-out partition is replaced with content from the replacement partition. This is repeated for a second read-out partition of the data object.

Independent **claim 41** is directed to a method for accessing data in a storage system. A first partition of a first data object in the storage system is obtained. A computation is performed using data comprising the first data object to produce a first computed value. Partition identification information relating to the first partition is obtained, the partition identification information including a first previously computed value. If the first computed value does not match the first previously computed value, then a first candidate partition is obtained and a second computed value is produced from the first candidate partition, the first candidate partition having a corresponding second previously computed value. If the second computed value does not match the second previously computed value, then repeating the foregoing with a second candidate partition. If the second computed value does match the second previously computed value, then the data constituting the first partition is replaced with data constituting the first candidate partition.

Independent **claim 46** is directed to a data storage system comprising a storage subsystem and a data processing subsystem. The data processing system accesses a first file stored on the storage subsystem. The data processing system determines, for each partition of the first file, whether the partition is corrupt. For each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed

partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

Independent **claim 50** is directed to a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 5,440,727 Bhide et al

The patent to Bhide et al. discloses asynchronous replica management in shared nothing architectures. In a partitioned database system of the Shared Nothing type, one or more secondary replicas of each partition are maintained by spooling (i.e., asynchronously sending) modified (usually called dirty) pages from the primary replica to the secondary replica(s) rather than by using a synchronous page update or by sending log entries instead of entire pages. A Write-Ahead Log protocol is used so that a dirty page is not forced to non-volatile storage until a log record of the modification is created and written to non-volatile storage. Replica updating does not delay the committing of transactions because replica updating is done asynchronously with respect to transaction processing. Since dirty pages are sent rather than only log entries, disk accesses and processing at the secondary replica(s) arising from the maintaining of the replicas are minimized as well. Only one centrally accessible log is maintained for all replicas of the same partition. When a failed system recovers from the failure, the replica(s) it maintains must be brought up to date. This may be done from the corresponding primary replica by sending dirty pages to the recovering system (assuming the corresponding primary replica is available and has maintained the needed dirty page table entries). Recovery may be done alternatively from the log in the event the corresponding primary replica is not available or has not maintained the needed dirty page table entries or because recovery from the log is preferred. (See, e.g., Abstract and column 4, lines 43-51).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 5,778,395 Whiting et al

The patent to Whiting et al. discloses a system for backing up files from disk volumes on multiple nodes of a computer network to a common random-access backup storage means. As part of the backup process, duplicate files (or portions of files) may be identified across nodes, so that only a single copy of the contents of the duplicate files (or portions thereof) is stored in the backup storage means. For each backup operation after the initial backup on a particular volume, only those files which have changed since the previous backup are actually read from the volume and stored on the backup storage means. In addition, differences between a file and its version in the previous backup may be computed so that only the changes to the file need to be written on the backup storage means. All of these enhancements significantly reduce both the amount of storage and the amount of network bandwidth required for performing the backup. Even when the backup data is stored on a shared-file server, data privacy can be maintained by encrypting each file using a key generated from a fingerprint of the file contents, so that only users who have a copy of the file are able to produce the encryption key and access the file contents. To view or restore files from a backup, a user may mount the backup set as a disk volume with a directory structure identical to that of the entire original disk volume at the time of the backup. When searching for matching files across users, it is usually deemed sufficient to have matching file size, file name, time/date, and hash value (e.g., CRC) computed over the file contents. While this approach does involve a finite (though minute) probability of false match, the error probability is acceptably small for almost all practical applications. In general, however, the file size (or at least some number of least significant bits of the size) and the hash value on the file contents are required to be equal in order for a file already in the database to be judged identical to a new/updated file being backed up. (See, e.g., Abstract and column 20, line 42-column 21, line 18).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach

or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 5,815,649 Utter et al

The patent to Utter et al. discloses a fault-tolerant computer system that comprises a plurality of processing nodes and a plurality of storage nodes interconnected by a network. The processing nodes perform processing operations in connection with user-generated processing requests. The processing nodes, in connection with processing a processing request, generate storage and retrieval requests for transmission to the storage node to enable storage of data thereon and retrieval of data therefrom. The storage nodes store data in at least one replicated partition group comprising a plurality of replicated partitions distributed across the storage nodes. A storage node, on receiving a retrieval request from a processing node provide the requested data to the processing node. In addition, on receiving a storage request from a processing node, a storage node initiates an update operation to update all of the replicated partitions in the replicated partition group. Following correction of a malfunction or failure of a storage node, partitions maintained by the malfunctioning or failed storage node can be recovered by use of the other members of the replicated partition group. The structure of the replication table, which identifies the various partitions comprising each of the replicated partition groups, is depicted in FIG. 4. With reference to FIG. 4, the replication table includes a number of entries each associated with a replicated partition group. Each entry includes a replicated partition group identifier field, and a number of sub-entries. (See, e.g., Abstract, column 6, line 62 and following, and column 12, lines 12-27).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third

predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed

value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 6,192,472 Garay et al

The patent to Garay et al. discloses a method and apparatus for the secure distributed storage and retrieval of information. A solution to the general problem of Secure Storage and Retrieval of Information (SSRI) guarantees that also the process of storing the information is correct even when some processors fail. A user interacts with the storage system by depositing a file and receiving a proof that the deposit was correctly executed. The user interacts with a single distinguished processor called the gateway. The mechanism enables storage in the presence of both inactive and maliciously active faults, while maintaining (asymptotical) space optimality. This mechanism is enhanced with the added requirement of

confidentiality of information; i.e., that a collusion of processors should not be able to learn anything about the information. Also, in this case space optimality is preserved. At the beginning of the refreshing phase, each server broadcasts to the other servers the fingerprints. Server Vi takes a majority vote among the received fingerprints to identify the correct ones. It then checks if its own fingerprints are correct. If they are corrupted, it replaces them with the correct ones. It then checks its own IDA-share Fi against the correct fingerprint {character pullout}(F1). If the share has been modified, it broadcasts a message asking the other servers to reconstruct Fi for it. It then takes a majority from among the received messages to identify the correct Fi. See, e.g., Abstract and column 15, lines 55-65).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement

partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 6,367,029 Mayhead et al

The patent to Mayhead et al. discloses a file server system tolerant to hardware and software failures is located over a plurality of hardware nodes. The nodes of the system act as hosts for software components of the system. Several of the software components can be replicated. The replicable software components include the system file store, a checker and a logger. The replicated components have one primary copy and one or more back-up copies. Replica copies of a given replicated component are each located at different nodes. Location and handling of replica copies of a given replicable component is under the control of a replication manager which is a (non-replicable) software component of the system. The replication manager is distributed meaning it can have one of its instances running on each node of the system. These instances inter-communicate to maintain coherence. The failure detector is also distributed, its instances running on each of the nodes, and contributing to an early detection of hardware and software failures. The file store is configured to hold stored objects and includes a signature generator for computing an object-specific signature from an object. The checker comprises a signature store for holding a previously computed signature for each of the stored objects and a comparator operable to compare a signature retrieved from the signature store with a corresponding signature computed by the signature generator from an object retrieved from the file store, thus to enhance system reliability. The object register 23 also stores, for each register

entry, information on any group affiliation of that hr-object. Object groups are used for replica copies of hr-objects. The group affiliation may be deleted or amended on request of the hr-object. Moreover, within a group, one of the replicas will be a primary and this will be registered with the replication manager. Change of the primary replica of the group (re-election) is also allowed for. (See, e.g., Abstract and column 7, lines 3-35).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a

partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 6,405,315 Burns et al

The patent to Burns et al. discloses a decentralized file system based on a network of remotely encrypted storage devices. The file system includes a network to which a network client, a secure remotely encrypted storage device, a key manager, and a lock manager are attached. The system organizes data as files and directories. Files or directories are composed of one or more streams, which logically partition the data associated with the files or directories. The device serves as a repository of the system's data. The key manager controls data access keys while the lock manager handles consistency of the files. A network user may have read or write access to a file. Access is controlled using keys and access lists maintained by the key manager. Files or directories are composed of one or more streams. Each stream logically partitions the data associated with the respective file or directory, and is physically composed of data objects on the device. Each directory data object includes an encrypted parent reference and multiple chunks. Each chunk is made up of file data together with a hash of the data that is then encrypted with an encryption key associated with the stream stripe set. The encrypted hash of the data allows detection of corruption or unauthorized changes of the chunks. The chunk size is used by the client to figure out the chunk boundaries to find the encrypted hash for integrity checking of the chunk. The chunks are stored on one or more file data objects. (See, e.g., Abstract, column 3, lines 32-43) and column 8, lines 5-20).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach

or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Patent No. 6,654,771 Parham et al

The patent to Parham et al. discloses a method and system for network data replication. Techniques are provided for replicating database changes among servers of a database system that effectively removes the inconsistency problems encountered after restoration from backup and due to removal and re-addition of partitions by assigning a new GUID to a server after it has been restored or after the partition has been re-added. This new GUID is used for identifying new changes made by the server after restoration or partition re-addition. By virtue of the use of the new GUID, new changes made after restoration will not be confused with any changes made by the server after the backup and before the restoration, which are identified by the old GUID of the server, thereby ensuring convergence of the servers' data through replication. Similarly, by virtue of the use of the new GUID, new changes made after re-addition of a partition will not be confused with any changes made prior to the re-addition of the partition. In another embodiment, the present invention is directed to a reliable and efficient way to handle replication among database servers when a partition has been removed from and re-added to a database. (See, e.g., Abstract and column 4, line 66-column 5, line 5).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the

data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Application Publication No. 2002/0032691 Rabii et al

The published patent application of Rabii et al. discloses a non-hierarchical or linear directory structure for a mass storage unit such as a disk. The directory structure can be kept in an auxiliary semiconductor memory. The disk is partitioned into segments of equal size. The directory structure presumes that data objects reside wholly and contiguously within a given area of the disk segments. While a variable number of objects may be stored within each segment, a given object is not allowed to occupy more than one segment. During a storage operation, objects are assigned to segments in a round-robin fashion, to equalize segment utilization. An object specifier for a requested object, such as in the form of the URL, is first converted or hashed to a predetermined but unique set of data bits. These bits are then used to locate a corresponding target object in the corresponding mass storage device. As will be

understood in further detail below, this permits lookup in an efficient fashion such as by using a first part of data bits to specify one of a set of lists or directory tables. The second portion of the bits can correspond to a particular entry in the table. Once the corresponding link list entry is determined, the corresponding target object address location associated with the requested object is then easily retrieved from the mass storage device. (See, e.g., Abstract and paragraph 36).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a

partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Application Publication No. 2002/0194209 Bolosky et al

The published patent application of Bolosky et al. discloses a file format for a serverless distributed file system that is composed of two parts: a primary data stream and a metadata stream. The data stream contains a file that is divided into multiple blocks. Each block is encrypted using a hash of the block as the encryption key. The metadata stream contains a header, a structure for indexing the encrypted blocks in the primary data stream, and some user information. The indexing structure defines leaf nodes for each of the blocks. Each leaf node consists of an access value used for decryption of the associated block and a verification value used to verify the encrypted block independently of other blocks. In one implementation, the access value is formed by hashing the file block and encrypting the resultant hash value using a randomly generated key. The key is then encrypted using the user's key as the encryption key. The verification value is formed by hashing the associated encrypted block using a one-way hash function. The file format supports verification of individual file blocks without knowledge of the randomly generated key or any user keys. To verify a block of the file, the file system traverses the tree to the appropriate leaf node associated with a target block to be verified. The file system hashes the target block and if the hash matches the access value contained in the leaf node, the block is authentic. To verify a block of the file, the file system optionally evaluates the signature on whole file verification value (if one exists), checks that the whole-file verification value matches the hash of the root block, metadata header and user information and then traverses the tree to the appropriate leaf node associated with a target block to be verified. (See, e.g., Abstract and paragraph 10).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more

replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage

system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

U.S. Application Publication No. 2003/0200207 Dickinson

The published patent application of Dickinson discloses methods and apparatuses for file synchronization and updating using a signature list. A server computer generates an update file for transmission to a client computer that permits the client computer to generate a copy of a current version of a subscription file from a copy of an earlier version of the subscription file. For each segment of the current version of the subscription file, the server computer searches an earlier version of a signature list for an old segment signature which matches a new segment signature corresponding to the segment. When a match is detected, the server computer writes a command in the update file for the client computer to copy an old segment of the client computer's copy of the earlier version of the subscription file into the client computer's copy of the current version of the subscription file, where the old segment corresponds to the segment for which a match was detected. When no match is detected, the server computer writes a command into the update file for the client computer to insert a new segment of the current version of the subscription file into the client computer's copy of the current version of the subscription file, where the new segment of the current version of the subscription file is written into the update file. The new segment of the current version of the subscription file may be compressed, encrypted, or both, by the server computer. When the update file is completed, the server computer transmits the update file to the client computer as an executable attachment via electronic mail. The update file is only generated when the server computer determines the subscription file has changed. The server computer periodically monitors the subscription file to determine if it has been altered before generating an update file. The signatures may be determined by any one of a variety of hashing methods or signature algorithms. In the presently preferred embodiment, the signatures are computed using the cyclic redundancy check (CRC). However, any signature algorithms may be used according to the invention. (See, e.g., Abstract and paragraph 52).

As to **claim 1**, the reference does not teach or suggest the method of accessing a storage system where a partition constituting a data object is processed to produce one or more replicas when it is determined that there are no other partitions among other data objects in the storage system that are identical to the partition of that data object.

As to **claim 8**, the reference does not teach or suggest, for a first partition of a file, that if the number of second partitions (defined as a partition comprising data belonging to a file in the storage system and is identical to the data of the first partition) is less than a first predetermined value, then a number of replicas of the first partition are produced sufficient to increase the number of second partitions to a second predetermined value. The reference does not teach or suggest that if the number of such second partitions is greater than a third predetermined value and if there are one or more replicas of the first partition, then one or more of the replicas are deleted. The reference does not teach or suggest repeating the foregoing for other partitions which constitute the first file.

As to **claim 18**, the reference does not teach or suggest identifying one or more partitions (referred to as input partitions) of a file, where for each input partition, if there are no identical partitions and if the number of replicas of the input partition is less than a threshold value, then at least one replica of the input partition is produced and stored on the storage system; where an identical partition is a partition, other than the partition, of a file that is stored in the storage system whose content is identical to content of the input partition.

As to **claim 28**, the reference does not teach or suggest a data storage system comprising a storage component and a data processing component; where the data processing component is configured to access a first partition of a file, and if the first partition does not have a corresponding identical partition in the storage component, then at least one replica is created. This is repeated for a second partition of the file.

As to **claim 33**, the reference does not teach or suggest a data processing system comprising means for producing a partition ID for each partition comprising a file, means for identifying one or more identical partitions among other files in the storage system based on a first partition ID, and means for creating a replica of the first partition in response to the second means making a determination that there are no identical partitions.

As to **claim 35**, the reference does not teach or suggest, for a first read-out partition of a data object, if content in the first read-out partition is corrupted, then the storage system is accessed to find a replacement partition from among one or more candidate partitions, including determining if a candidate partition is corrupted or not. The reference does not teach

or suggest that the content constituting the first read-out partition is replaced with content from the replacement partition, or repeating this for a second read-out partition of the data object.

As to **claim 41**, the reference does not teach or suggest, for a corrupted partition of a file, producing a computed value for a candidate partition and comparing the computed value against a previously computed value associated with the candidate partition. The reference does not teach or suggest that if the computed value does not match the previously computed value, then repeating the foregoing with a second candidate partition. The reference does not teach or suggest that if the computed value does match the previously computed value, then the data constituting the corrupted partition is replaced with data constituting the first candidate partition.

As to **claim 46**, the reference does not teach or suggest a data storage system comprising a storage subsystem and a data processing subsystem, where the data processing system determines, for each partition of a first file, whether the partition is corrupt. The reference does not teach or suggest that, for each corrupt partition, the data processing system determines whether there is a replacement partition on the storage system, the replacement partition being identical to the accessed partition at a time when the accessed partition was not corrupt, and modifies the first file to replace each of its corrupt partitions with a replacement partition if it exists.

As to **claim 50**, the reference does not teach or suggest a data storage system comprising means for accessing a partition comprising a file, means for determining whether a partition of a target file is corrupt, means for identifying a replacement partition from among a plurality of partitions stored on the storage subsystem to replace the corrupt partition based on a partition ID associated with the corrupt partition and on partition IDs of the plurality of partitions, and means to modify the target file to replace content comprising the corrupt partition with content from a replacement partition.

Conclusion

In view of this comments presented in the instant petition and the claim amendments presented in the accompanying preliminary amendment, the Examiner is respectfully requested to issue a first Office Action at an early date.

Respectfully submitted,

George B. F. Yee
Reg. No. 37,478

TOWNSEND and TOWNSEND and CREW LLP
Two Embarcadero Center, 8th Floor
San Francisco, California 94111-3834
Tel: 650-326-2400
Fax: 415-576-0300
Attachments
GBFY
60547378 v1